The 3-Coloring Problem

- Let $G = (V, E)$ be an undirected graph
- A 3-coloring of G assigns one of three colors to each vertex in V in such a way that no edge in E connects two vertices with the same color
- The search version of the 3-coloring problem asks us to find a 3-coloring of G, or determine that no such 3-coloring exists
- The decision version of the 3-coloring problem, denoted 3-COL, asks us to determine whether G admits a 3-coloring
- Exercise: Prove that if 3-COL is in P, then the search version of the 3-coloring problem can be solved in polynomial time

3-COL is NP-Complete

- Exercise: Describe a polynomial-time verifier for 3-COL
- We will prove that 3-SAT $\leq_{\mathrm{P}}$ 3-COL
  - Since 3-COL belongs to NP and 3-SAT is NP-complete, we conclude that 3-COL is NP-complete
- A significant fraction of NP-completeness proofs appearing in the literature are based on reductions from 3-SAT
  - The high-level structure of our "gadget-based" proof that 3-SAT $\leq_{\mathrm{P}}$ 3-COL is typical of such reductions

A "Gadget-Based" Transformation from 3-SAT to 3-COL

- Let f be a given 3-CNF formula
- We wish to transform (in polynomial time) f to a graph $G = (V, E)$ such that f is satisfiable if and only if G is 3-colorable
- The graph G is based on two kinds of "gadgets"
  - A gadget for each variable x in f that determines a truth assignment for x from any 3-coloring of G
  - For each clause C in f, there is a gadget that enforces the requirement that at least one literal in C is true

Inducing a Truth Assignment from a 3-Coloring

- Let us refer to the three colors as $\textsc{True}$ (true), $\textsc{False}$ (false), and $\textsc{Red}$ (red)
- For each variable x in f, we add two vertices x and x' to V
  - Vertex x is intended to represent the literal x
  - Vertex x' is intended to represent the literal $\neg x$
- How can we ensure that in any 3-coloring of G, either x gets color $\textsc{True}$ and x' gets color $\textsc{False}$, or x gets color $\textsc{False}$ and x' get color $\textsc{True}$?
  - Strictly speaking, this is impossible, since we can always permute the colors in any 3-coloring

Canonicalizing the Coloring

- We add three distinguished vertices called $\textsc{True}$, $\textsc{False}$, and $\textsc{Red}$ to V, and add edges $(\textsc{True}, \textsc{False})$, $(\textsc{True}, \textsc{Red})$, and $(\textsc{False}, \textsc{Red})$ to E
- Any 3-coloring has to assign distinct colors to these three vertices
- Without loss of generality, we can assume that vertex $\textsc{True}$ is colored $\textsc{True}$, vertex $\textsc{False}$ is colored $\textsc{False}$, and vertex $\textsc{Red}$ is colored $\textsc{Red}$

The Variable Gadget

- Recall that for any variable x in f, we have added vertices x and x' to V
- We wish to ensure that in any 3-coloring of G, either x gets color $\textsc{True}$ and x' gets color $\textsc{False}$, or x gets color $\textsc{False}$ and x' get color $\textsc{True}$
  - By adding edge (x, x') to E, we ensure that vertices x and x' are colored differently
  - By adding edges $(x, \textsc{Red})$ and $(x', \textsc{Red})$ to E, we ensure that neither x nor x' is colored $\textsc{Red}$

The "Core" of G

- Let n denote the number of variables in the given 3-CNF formula
- So far, we have added $2n + 3$ vertices to V and $3n + 3$ edges to E
- We refer to this subgraph of (the eventual) G as the core

The Clause Gadget: High-Level Plan

- For each clause C in f, we will add a "clause gadget" to G consisting of a set of vertices V_C and a set of edges E_C
  - Each edge in E_C will either have both endpoints in E_C or one endpoint in V_C and the other endpoint in the core
  - We refer to this as the isolation property of the clause gadgets
- We will design V_C and E_C so that a 3-coloring ϕ of the core can be extended to the clause gadget for C if and only if the truth assignment induced by ϕ satisfies clause C
  - Extending ϕ to the clause gadget for C means assigning colors to V_C such that u and v are assigned distinct colors for each edge (u, v) in E_C

The Clause Gadget: High-Level Plan (cont'd)

- Assume that we can design such a clause gadget
- Let σ be a satisfying truth assignment for f
  - σ corresponds to a specific 3-coloring ϕ' of the core of G
  - ϕ' can be extended to a 3-coloring ϕ that covers all of the clause gadgets (i.e., a 3-coloring of G), due to the isolation property
- Let ϕ be a 3-coloring of G
  - ϕ induces a coloring ϕ' of the core of G
  - ϕ' corresponds to a specific truth assignment σ to the variables of f
  - Since the 3-coloring ϕ' of the core can be extended to a 3-coloring of all the clause gadgets, σ is a satisfying assignment for f

Construction of V_C and E_C

- Let $C = \ell_1 \vee \ell_2 \vee \ell_3$ be a clause in f
- Our clause gadget is the union of three "chunks" and one "star"
- Let $V_C^{(i)}$ and $E_C^{(i)}$ denote the vertex and edge sets of chunk i for $1 \leq i \leq 3$
- Each set $V_C(i)$ will include a special vertex C_i
- For any chunk i, $1 \leq i \leq 3$, each edge in $E_C^{(i)}$ will either have both endpoints in $V_C(i)$, or will have one endpoint in $V_C(i)$ and one endpoint in the core
  - Thus, if we can extend a given 3-coloring ϕ of the core to any single chunk C_i in k_i different ways, then we can extend ϕ to $C_1 \cup C_2 \cup C_3$ in $\prod_{1 \leq i \leq 3} k_i$ ways

Construction of V_C and E_C (cont'd)

- We will design chunk 1 so that given any 3-coloring ϕ of the core
  - If ℓ_1 is colored $\textsc{False}$ in ϕ, then C_1 needs to be colored $\textsc{Red}$ in any valid extension of ϕ that colors chunk 1
  - If ℓ_1 is colored $\textsc{True}$ in ϕ, then there are valid extensions of ϕ that color chunk 1 and color C_1 differently
- We will design chunk 2 (resp., 3) to satisfy the same properties except that when ℓ_2 (resp., ℓ_3) is colored $\textsc{False}$, then C_2 has to be colored $\textsc{True}$ (resp., $\textsc{False}$)
- Assuming that we can achieve these properties, how should we construct the final "star" component of the clause gadget for C?

Construction of the "Star" Component

- The star component consists of a vertex C_0 and the three edges (C_0, C_1), (C_0, C_2), and (C_0, C_3)
- Let ϕ be a 3-coloring of the core such that every literal in C is colored $\textsc{False}$
  - The properties of the chunks ensure that C_1, C_2, and C_3 are assigned distinct colors in any valid extension of ϕ that colors the three chunks
  - Thus no valid extension of ϕ colors the entire clause gadget

Construction of the "Star" Component (cont'd)

- Let ϕ be a 3-coloring of the core such that at least one literal in C is colored $\textsc{True}$
  - The properties of the chunks ensure that some valid extension of ϕ colors the three chunks and uses at most two colors to color vertices C_1, C_2, and C_3
  - Thus there is a valid extension of ϕ that colors the entire clause gadget

What's Left?

- We just need to show how to construct chunks 1, 2, and 3 with the desired properties

Construction of Chunk 1

- We can set $V_C^{(1)}$ to $\{C_1\}$ and $E_C^{(1)}$ to

$$\{(C_1, \ell_1), (C_1, \textsc{True})\}$$

- If ℓ_1 is colored $\textsc{False}$, then C_1 has to be colored $\textsc{Red}$
- If ℓ_1 is colored $\textsc{True}$, then C_1 can be colored $\textsc{Red}$ or $\textsc{False}$

Construction of Chunk 2

- We can set $V_C^{(2)}$ to $\{C_2, C_4\}$ and $E_C^{(2)}$ to

$$\{(C_2, C_4), (C_2, \textsc{False}), (C_4, \ell_2), (C_4, \textsc{True})\}$$

- If ℓ_2 is colored $\textsc{False}$, then C_4 has to be colored $\textsc{Red}$ and hence C_2 has to be colored $\textsc{True}$
- If ℓ_2 is colored $\textsc{True}$, then C_4 can be colored $\textsc{Red}$ or $\textsc{False}$ and hence C_2 can be colored $\textsc{True}$ or $\textsc{Red}$

Construction of Chunk 3

- We can set $V_C^{(3)}$ to $\{C_3, C_5\}$ and $E_C^{(3)}$ to

$$\{(C_3, C_5), (C_3, \textsc{True}), (C_5, \ell_3), (C_5, \textsc{True})\}$$

- If ℓ_3 is colored $\textsc{False}$, then C_5 has to be colored $\textsc{Red}$ and hence C_3 has to be colored $\textsc{False}$
- If ℓ_3 is colored $\textsc{True}$, then C_5 can be colored $\textsc{Red}$ or $\textsc{False}$ and hence C_3 can be colored $\textsc{False}$ or $\textsc{Red}$

Some Complexity Classes beyond P

- Thus far we have discussed the classes P and NP, as well as the class of NP-complete languages
- The class co-NP is defined as the set of all languages L such that the complement of L (i.e., the set of all strings not in L) belongs to NP
- It is widely believed that $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{NP} \neq \text{co-NP}$
  - If $\mathrm{P} = \mathrm{NP}$ then $\mathrm{NP} = \text{co-NP}$, but the converse does not necessarily hold

NP-Intermediate

- Languages that are in NP and not in P are called NP-intermediate
  - Of course, this class is empty if $\mathrm{NP} = \mathrm{P}$
- Candidate NP-intermediate languages include languages related to cryptographic problems like factoring and discrete log
- Another well-known candidate is graph isomorphism
  - In a recent breakthrough result, Babai presented an algorithm with running time $\exp((\log n)^{O(1)})$

The Polynomial Hierarchy

- A canonical NP-complete language is satisfiability, which asks whether $\exists x_1 \ldots \exists x_n f(x_1, \ldots, x_n)$ where f is a propositional formula in the variables $x_1, \ldots, x_n$
- Likewise, a canonical co-NP-complete language is $\forall x_1 \ldots \forall x_n f(x_1, \ldots, x_n)$?
- By introducing more alternations, we get other classes
  - The class where we start with existential quantification and have k levels of quantifiers is sometimes denoted Σ_k
  - The class where we start with universal quantification and have k levels of quantifiers is sometimes referred to as Π_k
- The "polynomial hierarchy" refers to the collection of all of these classes, over all k

PSPACE and Beyond

- PSPACE is the class of all languages decidable in polynomial space
- A canonical complete language for PSPACE is $\exists x_1 \forall x_2 \ldots \forall x_{2n} f(x_1, \ldots, x_{2n})$
- It is an open question whether $\mathrm{P} = \mathrm{PSPACE}$
- There are many other complexity classes beyond PSPACE
    - For example, EXPTIME is the class of all languages that can be decided in exponential time
    - It is easy to argue that EXPTIME contains PSPACE
    - It is an open question whether EXPTIME is equal to PSPACE
    - The "time hierarchy theorem" implies that EXPTIME properly contains P